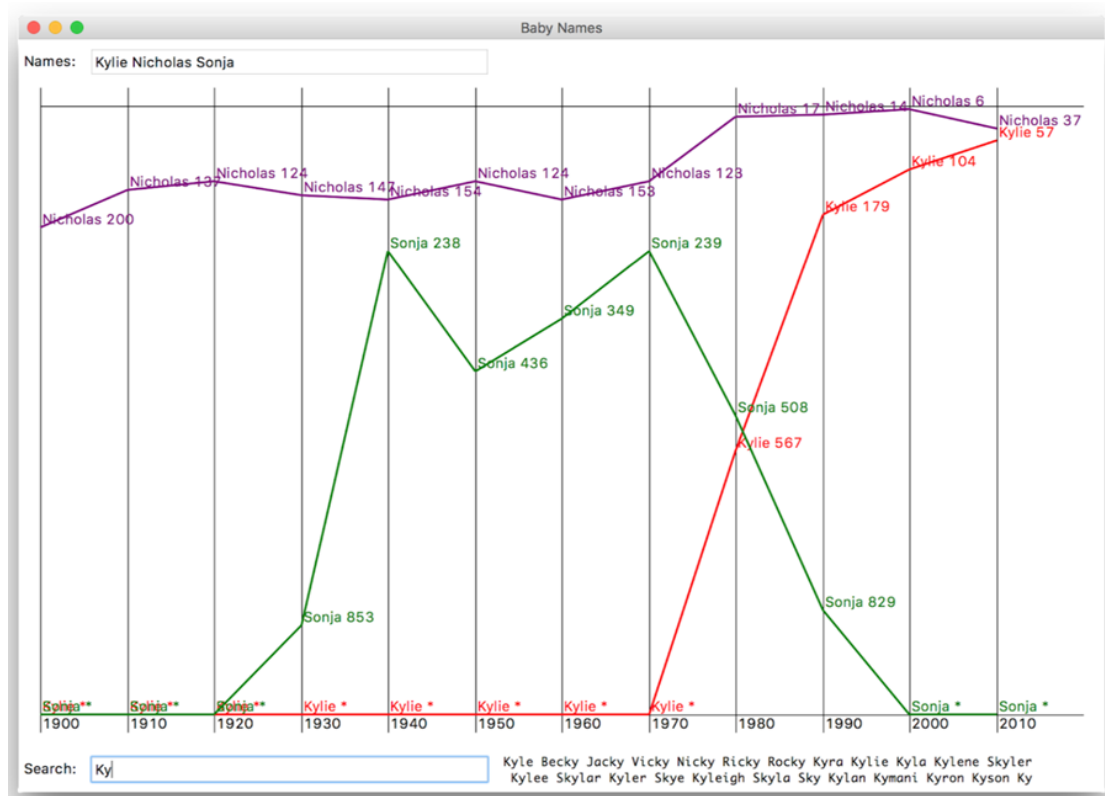


stanCode

標準程式教育機構

Assignment 4

This assignment is based on the Assignment 4 of CS106P at Stanford University



作業檔案下載

歡迎來到 **SC101** 大數據處理！我們從兩個月前第一次接觸 **Python** 到現在已經具備足夠的能力成為一位軟體工程師了！希望同學為 **stanCode** 作業苦思、熬夜、備感壓力的同時也得到了相對應 **coding** 技能提升的回饋！

作業預計時間：15 小時

如果作業卡關 歡迎各位到社團提問，也非常鼓勵同學們互相討論作業之概念，但請勿把 code 給任何人看（也不要將程式碼貼在社團裡）分享妳/你的 code 會剝奪其他學生獨立思考的機會，也會讓其他學生的程式碼與你/妳的極度相似，使防抄襲軟體認定有抄襲嫌疑

Baby Names

Baby Names 是一個整理 1900 - 2010 所有新生兒英文名字並繪製圖表的程式

這份作業內容與一位資料科學家的日常非常相似。請同學務必妥善安排時間一步一步完成本次作業的 **milestones**，並於開始這份作業前熟悉上課使用的資料結構觀念

請務必細讀作業裡的每一行文字

每一年美國社會安全局都會在網站 (<http://www.ssa.gov/OACT/babynames>) 上公佈最「夯」的 **Top 1000** 新生兒姓名。網站上的資料呈現如下圖所示：

Rank	Male name	Female name
1	Jacob	Emily
2	Michael	Hannah
3	Matthew	Madison
4	Joshua	Ashley
5	Christopher	Sarah
...		

Figure 1. 西元 2000 年新生兒姓名資料 (由左至右依序為：排名、男生名、女生名)

Rank 1 代表當年度最夯的名字；超過 **Rank 1000** 以後的名字就沒有記錄了。我們已經將網站上的資料整理成如下圖所示的文字檔，同學們可以在 **data/full** 資料夾中找到從西元 1990 到 2010 年的所有資料

baby-1980.txt	baby-2000.txt
<pre> 1980 1,Michael, Jennifer 2,Christopher,Amanda 3, Jason,Jessica 4,David,Melissa 5,James, Sarah . . . 780,Jessica,Juliana 781, Mikel, Charissa 782,Osvaldo,Fatima 783,Edwardo,Shara 784, Raymundo, Corrie . . . </pre>	<pre> 2000 1,Jacob, Emily 2, Michael, Hannah 3, Matthew,Madison 4, Joshua, Ashley 5,Christopher,Sarah . . . 240, Marcos,Gianna 241,Cooper, Juliana 242, Elias,Fatima 243,Brenden,Allyson 244,Israel, Gracie . . . </pre>

Figure 2. 西元 1980 年 (左) 與西元 2000 年 (右) 新生兒姓名資料片段

Figure 2. 可以看到很有趣的現象：1980 年代時 Jennifer 是女性名字裡最有名的，然而，到了 2000 年竟滑出滑出五名之外；在 1980 年代鮮少出現的 Juliana (#780) 到了 2000 年竟大躍進了 300 多名！ (#241)

「新生兒名字名氣隨時代起伏的趨勢」將是這份作業要請同學完成的工作。我們把這份巨型工作拆解成了六個 milestones，下面將一一解說：

Milestone 1 - milestone1.py

```
add_data_for_name(name_data, year, rank, name)
```

第一部分要請同學編輯 milestone1.py (不是 babynome.py) 並完成裡面的 add_data_for_name

- Parameters 介紹

1. name_data(dict): key 是新生兒的名字；value 是另一個 dictionary，裝著 name 在 year 的 rank。(請原諒我們的中英混雜...)
2. year(str): 代表某一個年代
3. rank(str): 代表該新生兒名字在當年的排名
4. name(str): 新生兒名字

- 此 function 沒有 return 任何東西

假設 **name_data** 在進入 **add_data_for_name** 之前的狀態為：

```
{ "Kylie": {"2010": "57"}, "Nick": {"2010": "37"} }
```

經過 **add_data_for_name(name_data, "2010", "208", "Kate")** , **name_data** 會變成 { "Kylie": {"2010": "57"}, "Nick": {"2010": "37"}, "Kate": {"2010": "208"} }

如下圖所示：

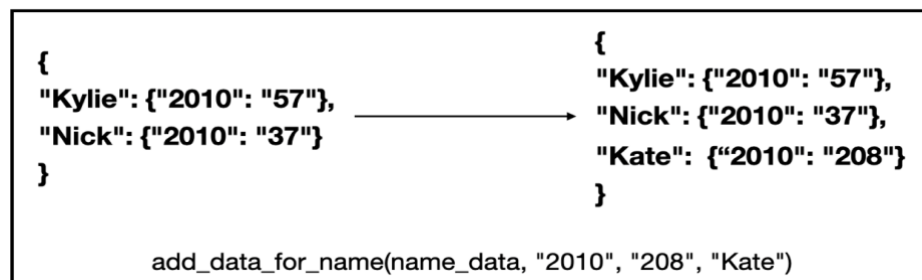


Figure 3. **name_data** 進入 **add_data_for_name** 之前 (左圖) 之後 (右圖)

Name in different years

若同一個名字在不同年份有相對應的排名，請同學將新資料加到 **name_data** 中。
舉例來說，假設 **name_data** 進入 **add_data_for_name** 之前的狀態為：

```
{ "Kylie": {"2010": "57"}, "Nick": {"2010": "37"} }
```

經過 **add_data_for_name(name_data, "2000", "104", "Kylie")** , **name_data** 會變成 { "Kylie": {"2010": "57", "2000": "104"}, "Nick": {"2010": "37"} } 如下圖

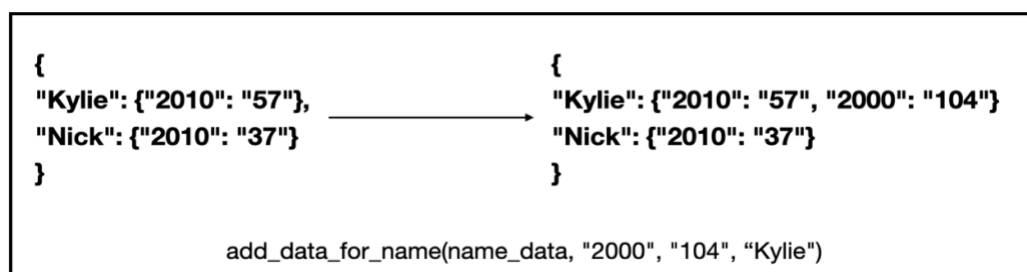


Figure 4. **name_data** 進入 **add_data_for_name** 之前 (左圖) 之後 (右圖)

男 Sammy 與女 Sammy

有些中性名字，如 **Sammy**，可能在某一年同時出現在男、女姓名排行榜。因此，請同學在加入一個新名字 **name**, **year** 進入 **name_data** 前，先檢查 **name**, **year** 是否已存在於 **name_data**；若已經存在，請把 **name** 在 **year** 的 **rank** 儲存「比較低的排名」。舉例來說，若 **Sammy** 在 "1990" 有排名 "90" 以及 "200"，請同學儲存 **name = Sammy** ; **year = "1990"**; **rank = "90"**

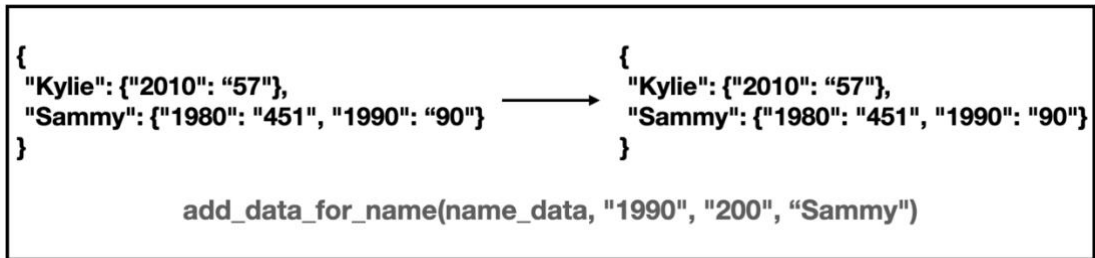


Figure 5. `name_data` 進入 `add_data_for_name` 之前 (左圖) 之後 (右圖)

測試方法

Mac 請在 terminal 輸入 **python3 milestone1.py test1**

Windows 請在 terminal 輸入 **py milestone1.py test1**

- 若您程式正確，您將看到下列文字

```
-----test1-----
{'Kylie': {'2010': '57'}, 'Nick': {'2010': '37'}, 'Kate': {'2010': '208'}}
-----
```

Mac 請在 terminal 輸入 **python3 milestone1.py test2**

Windows 請在 terminal 輸入 **py milestone1.py test2**

- 這個將會測試您的程式是否可以通過 **Name in different years** 的測試
- 若程式正確，您將看到下列文字

```
-----test2-----
{'Kylie': {'2010': '57', '2000': '104'}, 'Nick': {'2010': '37'}}
-----
```

Mac 請在 terminal 輸入 **python3 milestone1.py test3**

Windows 請在 terminal 輸入 **py milestone1.py test3**

- 這個將會測試您的程式是否可以通過 男 Sammy 與女 Sammy 的測試
- 若程式正確，您將看到下列文字

```
-----test3-----
{'Kylie': {'2010': '57'}, 'Sammy': {'1980': '451', '1990': '200'}, 'Kate': {'2000': '20'}}
```

Putting It All Together

最後，我們整理了一些學生常犯的錯誤讓各位測試 Milestone 1, 以確保 Milestone 2 - 6 都可以順利執行

測試方法

Mac 請在 terminal 輸入 **python3 milestone1.py test4**

Windows 請在 terminal 輸入 **py milestone1.py test4**

- 若您程式正確，您將看到下列文字

```
-----test4-----
{'Kylie': {'2010': '57', '2000': '104'}, 'Nick': {'2010': '37'},
'Kate': {'2010': '208', '2000': '108'}, 'Sammy': {'1990': '90'}}
```

Milestone 2 - babyname.py

add_file(name_data, filename)

- Parameters 介紹
 1. **name_data(dict)**: **key** 是新生兒的名字；**value** 是另一個 **dictionary** (裝著 **name** 在 **year** 的 **rank**)
 2. **filename(str)**: 某一年的文字檔檔名
- 此 function 沒有 return 任何東西

確定 **Milestone 1** 都通過測試後，同學可以把 `add_data_for_name` 的程式碼從 `milestone1.py` 複製到 `babyname.py` 的 `add_data_for_name` 裡面

再來先請同學編輯名為 `add_file(name_data, filename)` 的程式。`add_file` 會把檔名為 `filename` 的文字檔所有資料加到 `name_data` 裡。請同學在編輯 `add_file` 時使用您於 **Milestone 1** 建造的 `add_data_for_name`

所有您要處理的文字檔大致都與第三頁的 **Figure 2** 大同小異：檔案開頭的第一行為 `year`，緊接著是 `rank` 從 1 到 1000 的男生姓名 (`name1`) 與女生姓名 (`name2`)

請注意：文字檔中 `rank` 與 `name1` 中間一定有一個逗號、`name1` 與 `name2` 間也一定有一個逗號。然而，`rank`, `name1`, `name2` 的前後可能會有不固定的空白，因此請同學在將 `rank`, `name1`, `name2` 輸入到 `add_data_for_name` 之前，請先做 `rank=rank.strip()`, `name1=name1.strip()`, `name2=name2.strip()`

`strip()` 為 `string` 內建 `method`，可以把一個文字前後所有空白去除。舉例來說，`name1 = ' Jerry \n'`；經過 `name1 = name1.strip()` 後，`name1` 就會變成 `'Jerry'`

Milestone 2 將與 Milestone 3 一併測試

Milestone 3 - babyname.py

`read_files(filenamees)`

- **Parameters**
 1. `filenamees(list)`: 裝著許多檔名的 **Python List**
- **return** 一個 **Python Dictionary** (就是 `name_data`)

`search_names(name_data, target)`

- **Parameters**
 1. `name_data(dict)`: 裝著新生兒資訊的 **Python Dictionary**
 2. `target(str)`: 文字片段
- **return** 一個 **Python List** 裝著所有包含 `target` 的名字

完成 **Milestone 2** 之後，請編輯 `read_files(filenamees)`。`filenamees` 為一個 `list`，裡面的每一個元素都是一個檔案名稱 (`filename`)。因此，請使用 **Milestone 2** 建造的 `add_file(name_data, filename)` 將名為 `filename` 檔案的資料加入 `name_data` 裡

所有在 `filenamees` 裡的檔案都被讀取與處理後，請同學 `return name_data`

完成上述要求後，請同學再前往編輯 `search_names(name_data, target)`。`target` 為某個名字的片段，您將搜尋所有在 `name_data` 裡包含 `target` 的所有名字，並將它們儲存在一個名叫 `names` 的 `Python List`，最後 `return names`。這邊要請同學們特別注意的是，`search_names` 應該為 `case-insensitive`。舉例來說，若 `target == 'aa'`，您的 `names` 裡面應該也要包含 “Aaron” 這個名字

測試方法

- Mac 請在 `terminal` 輸入
`python3 babynames.py data/small/small-2000.txt data/small/small-2010.txt`
- Windows 請在 `terminal` 輸入
`py babynames.py data/small/small-2000.txt data/small/small-2010.txt`
- 這個指令將測試您 `add_file(...)` 與 `read_files(...)` 的程式是否通過基本要求

- 如果您的程式正確，應該會看到下方文字


```
A [('2000', '1'), ('2010', '2')]
B [('2000', '1')]
C [('2000', '2'), ('2010', '1')]
D [('2010', '1')]
E [('2010', '2')]
```

- Mac 請在 `terminal` 輸入
`python3 babynames.py -search aa data/full/baby-2000.txt data/full/baby-2010.txt (不需換行)`

Windows 請在 `terminal` 輸入
`py babynames.py -search aa data/full/baby-2000.txt data/full/baby-2010.txt (不需換行)`

- 這個指令將測試您 `search_names` 的程式是否通過基本要求
- 如果您的程式正確，應該會看到下方文字

```
Aaron  
Isaac  
Aaliyah  
Isaak  
Aaden  
Aarav  
Ayaan  
Sanaa  
Ishaan  
Aarush
```



Milestone 4 - babygraphics.py

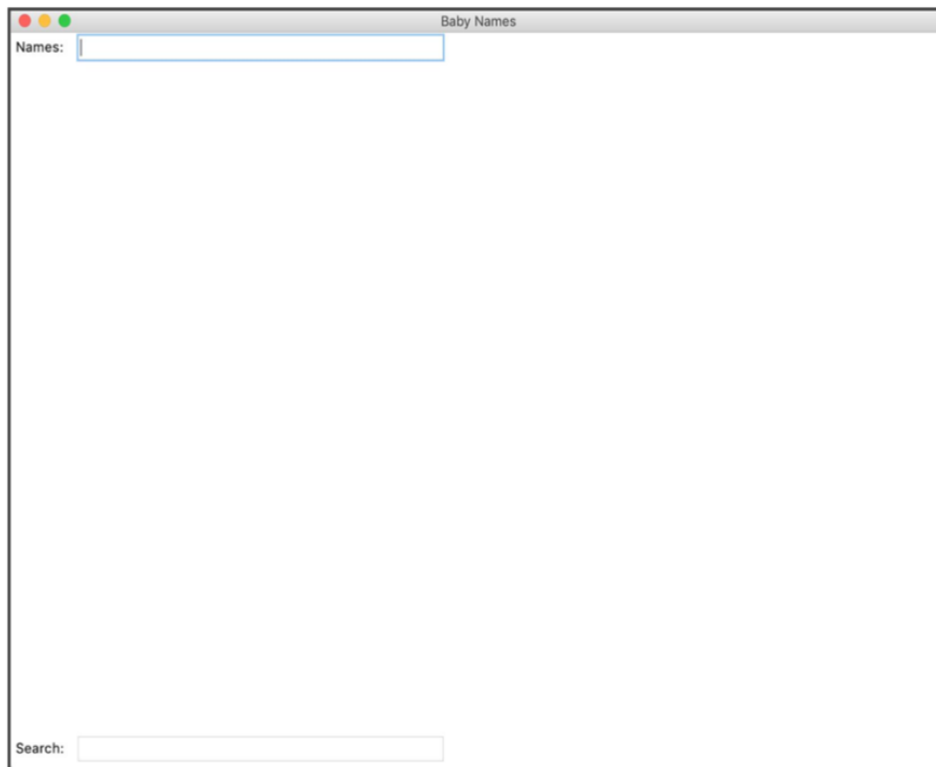
恭喜同學完成一半囉！**Milestone 1-3** 就是一位後端工程師的工作內容。再來 **Milestone 4-6** 要請同學轉換成一位前端工程師！

首先，

Mac 請在 **terminal** 輸入 `python3 babygraphics.py`

Windows 請在 **terminal** 輸入 `py babygraphics.py`

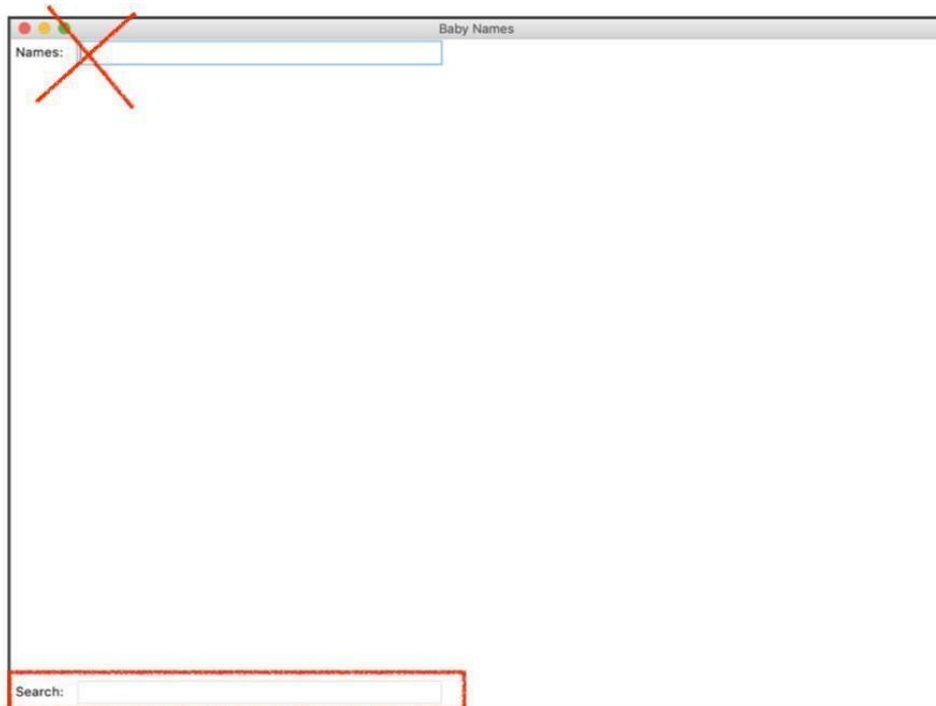
執行後應該會看到一個視窗（我們已經幫各位完成一半的前端工程，再來交給各位了！）



請在視窗左下方 **Search:** 裡輸入 **aa** 比對視窗右下方出現的文字與下圖是否一致



請注意：搜尋欄有兩個。**Search** 是在左下角，不是左上角



如果沒什麼問題，**Milestone 4** 就完成囉！

Milestone 5 - babygraphics.py

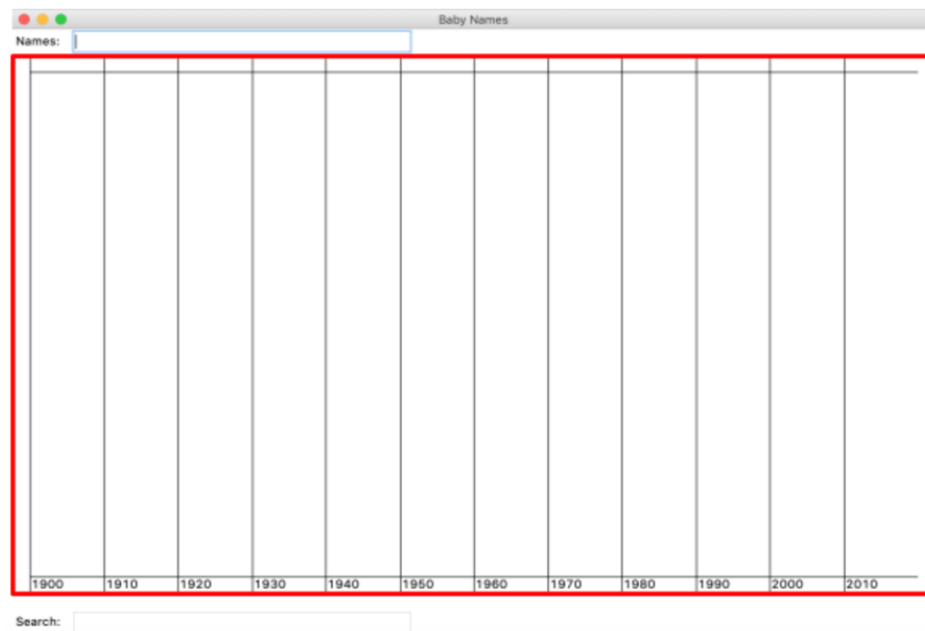
`draw_fixed_lines(canvas)`

- Parameters

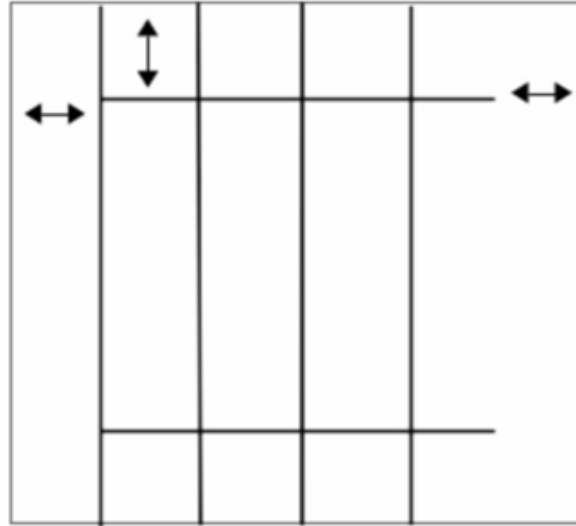
1. `canvas` - `tkinter.Canvas` - 視窗，可使用 `canvas.create_line(...)` 以及 `canvas.create_text(...)`

- 不需要 `return` 任何東西

下圖用紅線匡起來的區間我們稱之為 **canvas**



第一步，請同學先在您空白的 **canvas** 上加兩條橫線：一條與底端距離 **GRAPH_MARGIN_SIZE**；另一條與頂端距離 **GRAPH_MARGIN_SIZE**；再來請同學在 **canvas** 左、右兩邊留白，距離 **GRAPH_MARGIN_SIZE** (如下圖所示)



Constant 介紹

- **FILENAMES** 為所有資料的檔名 (您不會使用到此常數)
- **CANVAS_WIDTH** 為視窗的寬
- **CANVAS_HEIGHT** 為視窗的高
- **YEARS** 為一個 **Python List**，裡面的每一個元素為我們想要分析的年份 (長度不一定為 12)
- **GRAPH_MARGIN_SIZE** 為線與視窗的保留距離
- **COLORS** 為您繪製圖表之顏色出現順序
- **TEXT_DX** 為文字左邊與直線之間的距離
- **LINE_WIDTH** 為每一條線的寬度
- **MAX_RANK** 為排名最後一個編號

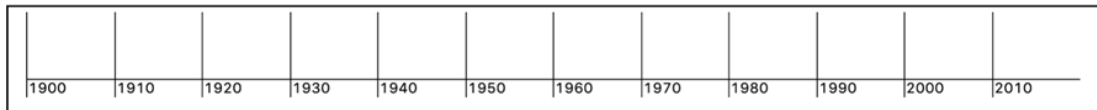
再來我們要按照 **YEARS** 裡的資料將 **canvas** 從 **GRAPH_MARGIN_SIZE** 到 **width - GRAPH_MARGIN_SIZE** 用直線等距分割。每一條直線由上到下的長度為 **height**，但 **x** 座標會隨著資料在 **YEARS** 的位置不同而改變。因此，請同學完成 **get_x_coordinate(width, year_index)**

get_x_coordinate(width, year_index)

- **Parameters**
 1. **width(int)**: 就是 **CANVAS_WIDTH**
 2. **year_index(int)**: 在 **YEARS** 這個 list 裡面所屬的 **index**
- **return** 一個整數，為 **year_index** 在 **canvas** 的 **x** 座標

舉例來說，若 `YEARS = ['1900', '1910', '1920', '1930', '1940', '1950', '1960', '1970', '1980', '1990', '2000', '2010']` 則 `get_x_coordinate(1000, 2)` 應該 `return` 一個整數為 **180** (代表 **1920** 線的 x 座標)

請使用 `get_x_coordinate` 在 `canvas` 加上許多直線。每一條直線與 `canvas` 底部橫線都有一個交叉點，請在此交叉點的右邊 `TEXT_DX` 距離加上此直線所代表的年份 (如下圖所示)



提示：`canvas.create_text(...)` 使用 `anchor = tkinter.NW` 會把文字的西北方當成基準點

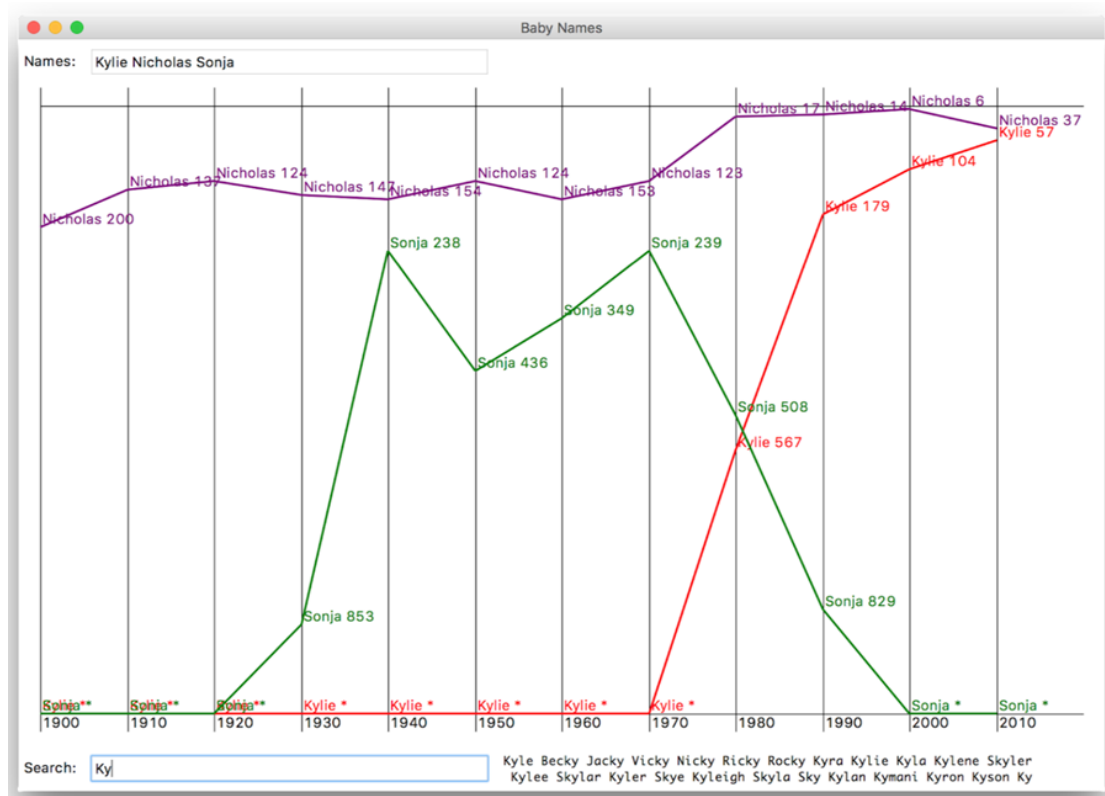


Milestone 6 - babygraphics.py

`draw_names(canvas, name_data, lookup_names)`

- **Parameters**
 1. `canvas(tkinter.Canvas)`: 視窗，可使用 `canvas.create_line(...)` 以及 `canvas.create_text(...)`
 2. `name_data(dict)`: 裝著所有新生兒資訊的 **Python Dictionary**
 3. `lookup_names(list)`: 裝著使用者要查看名字的 **Python List**
- 不需要 `return` 任何東西

本次作業的最後一個部分要請同學將 `lookup_names` 裡面的所有名字的資料畫在 `canvas` 上



若使用者在 **canvas** 左上方 **Names:** 後面輸入 **Kylie Nicholas Sonja** ,
lookup_names 就會是 ['Kylie', 'Nicholas', 'Sonja'] (這個部分我們已經寫好了)

每次只要使用者在視窗 **Names:** 後面的文字方格按下鍵盤上的 **<ENTER/return>** ,
draw_names(...) 就會被呼叫 (鍵盤的部分我們已經幫同學們寫好了)

如上圖所示，在 **1970** 前，**Kylie** 這個名字的排名在 **1000** 名之外。若某名字排名在 **1000** 名之外，請將此資料的排名以 **" * "** 代替

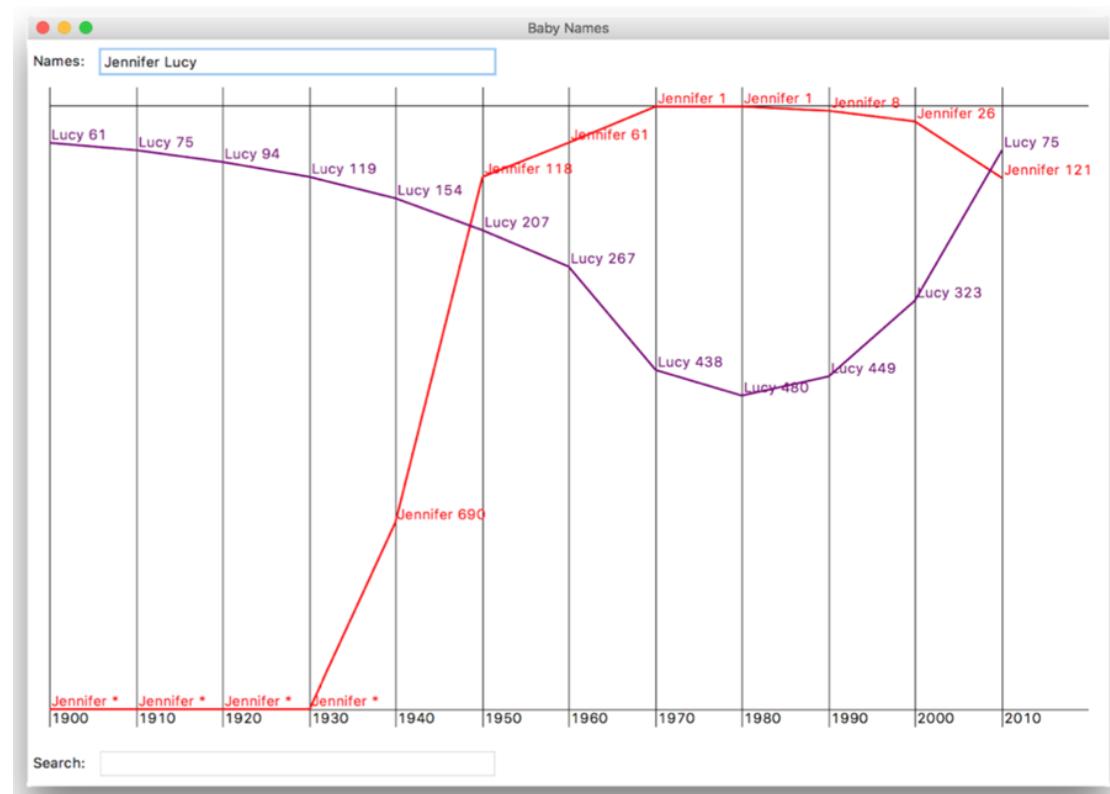
每一筆資料都會有文字 **"名字 排名"** (**Nicholas 200**) ，但**"名字 排名"**加上 **canvas** 的 **x** 座標應與直線距離 **TEXT_DX**

曲線顏色順序已經定義在名為 **COLORS** 的 **Python List** 裡。若繪製的數目大於 **COLORS** 的長度，請回到 **COLORS[0]** 從頭開始 (要改變線條為藍色，只要在 **canvas.create_line (...)** 括弧裡面加上 **fill = 'blue'** 即可)

曲線的寬度請定義為 **LINE_WIDTH** (要改變線條粗細為 **LINE_WIDTH** ，只要在 **canvas.create_line (...)** 括弧裡面加上 **width = LINE_WIDTH** 即可)

測試方法

Jennifer, Lucy 這兩個名字包涵了所有繪圖的情況。請在視窗上方輸入這兩個名字。若與下圖結果一致，**恭喜！** 您已經是一位**全端工程師** - 前端後端一手包辦的強者了



Extension - extension.py

我們在上課學會了如何使用 requests, bs4 的套件去抓取 imdb 電影數據。因此，這個部分要同學們自己嘗試從 US Social Security 網站抓取 babynames 資料！

下圖是 US Social Security 網站在 2010s 的資料（該網站目前僅提供前 200 名的男生名與女生名）

Top names of the 2010s

Popular Baby names

Select another decade?

2010s Go

Top names in last 100 years

Top 5 names in each year

Popular Names by State

The following table shows the 200 most popular given names for male and female babies born during the 2010s. For each rank and sex, the table shows the name and the number of occurrences of that name. The 200 most popular names were taken from a universe that includes 20,072,102 male births and 19,166,105 female births

Popular names of the period 2010s

Rank	Males		Females	
	Name	Number	Name	Number
1	Noah	182,993	Emma	194,755
2	Liam	173,717	Olivia	184,291
3	Jacob	162,958	Sophia	180,896
4	William	159,697	Isabella	170,265
5	Mason	157,652	Ava	155,606
6	Ethan	148,908	Mia	128,881
7	Michael	144,824	Abigail	118,526

若同學點選左邊下拉欄位 **2000s** 並按下 **Go**，您會發現 2000s 資料所在之網址跟 2010s 資料的網址幾乎一模一樣！只是結尾從 2010s.html 變成了 2000s.html (如下圖紅色圈圈所示)

Top names of the 2000s

Popular Baby names

Select another decade?

2000s Go

Top names in last 100 years

Top 5 names in each year

Popular Names by State

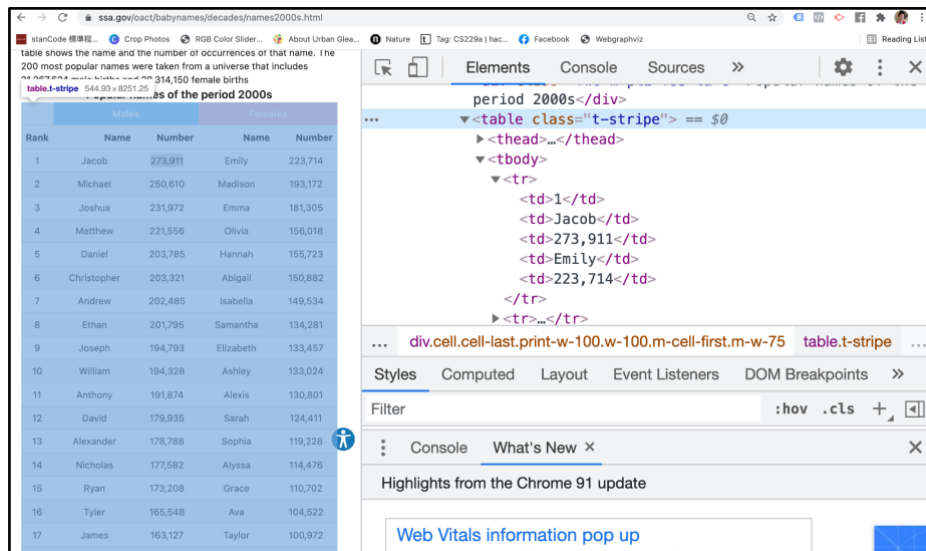
The following table shows the 200 most popular given names for male and female babies born during the 2000s. For each rank and sex, the table shows the name and the number of occurrences of that name. The 200 most popular names were taken from a universe that includes 21,267,624 male births and 20,314,150 female births

Popular names of the period 2000s

Rank	Males		Females	
	Name	Number	Name	Number
1	Jacob	273,911	Emily	223,714
2	Michael	250,610	Madison	193,172
3	Joshua	231,972	Emma	181,305
4	Matthew	221,556	Olivia	156,018
5	Daniel	203,785	Hannah	155,723
6	Christopher	203,321	Abigail	150,882
7	Andrew	202,485	Isabella	149,534

這個部分要同學嘗試將 2010s, 2000s, 與 1990s 的男、女新生兒前 200 名的人數總和計算出來 (也就是表格裡 Number 欄位的數字總和)

若您查看該網站的 HTML 會發現我們要的資料在 `<table class="t-stripe">` 裡面的 `<tbody>`：



提示：`<tbody>` 底下有許許多多的 `<tr>`，而 `<tr>` 裡面又有許許多多的 `<td>`
 若要得到所有 `<td>` 與 `</td>` 之間的文字，可以直接對 soup 裡面的 `<tbody>` 物件
 取出文字

請編輯 `extension.py` 檔案，並完成上述工作！若您所撰寫的程式正確，應該會在
 執行 `extension.py` 看到 Console 上的輸出與下圖一至 (執行時間大約 1 分鐘)：

```

-----
2010s
Male Number: 10890537
Female Number: 7939153
-----

2000s
Male Number: 12975692
Female Number: 9207577
-----

1990s
Male Number: 14145431
Female Number: 10644002

Process finished with exit code 0

```

評分標準

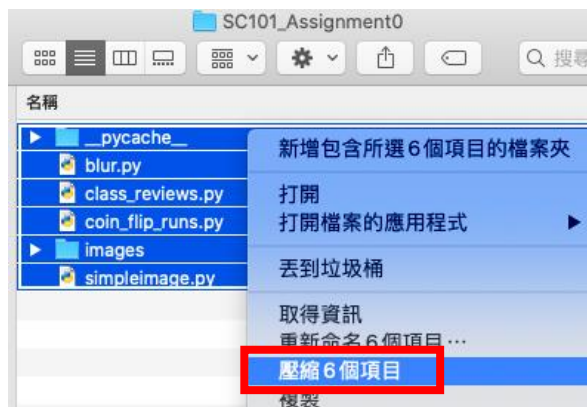
Functionality - 程式是否有通過我們的基本要求？程式必須沒有 bug、能順利完成指定的任務、並確保程式沒有卡在任何的無限環圈 (Infinite loop) 之中

Style - 好的程式要有好的使用說明，也要讓人一目瞭然，這樣全世界的人才能使用各位的 code 去建造更多更巨大更有趣的程式。因此請大家寫**精簡扼要**的使用說明、function 敘述、單行註解

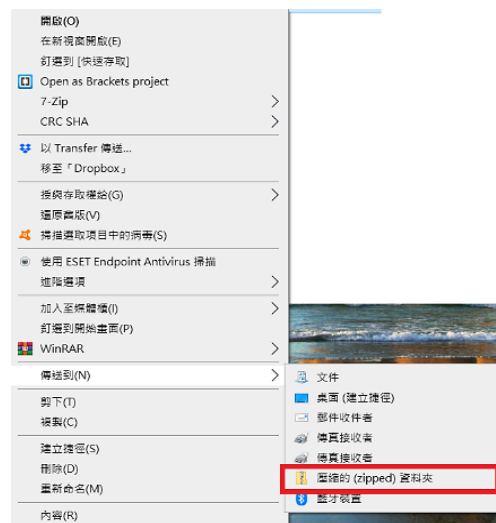
作業繳交

1. 以滑鼠「全選」作業資料夾內的所有檔案，並壓縮檔案。請見下圖說明。

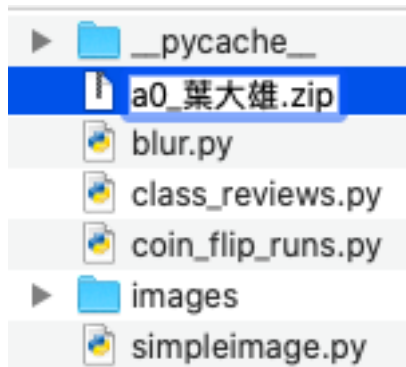
macOS：按右鍵選擇「壓縮 n 個項目」



Windows：按右鍵選擇「傳送到」→「壓縮的(zipped)資料夾」

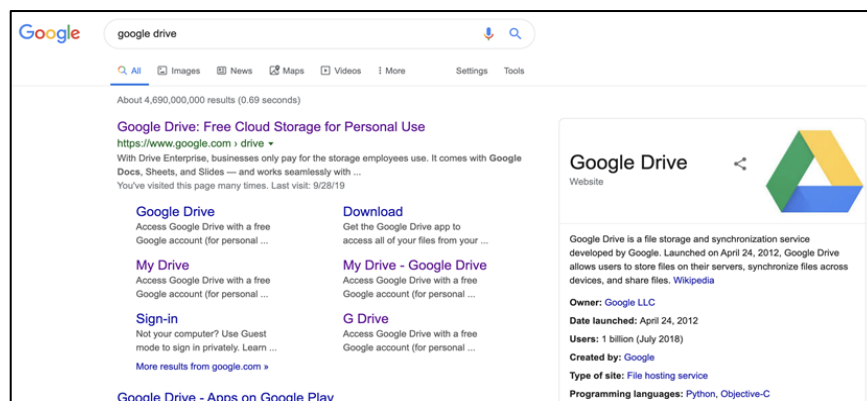


2. 將壓縮檔(.zip)重新命名為「a(n)_中文姓名」。如：
assignment 0 命名為 a0_中文姓名;
assignment 1 命名為 a1_中文姓名; ...



3. 將命名好的壓縮檔(.zip)上傳至 Google Drive (或任何雲端空間)

- 1) 搜尋「google drive」

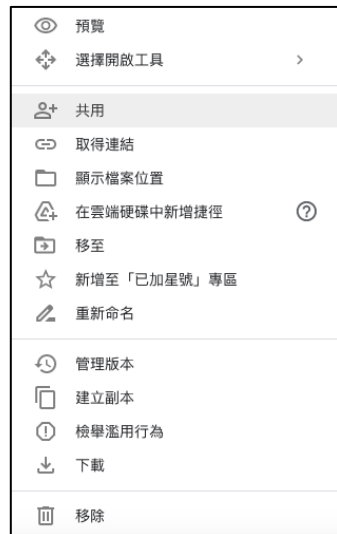


- 2) 登入後，點選左上角「新增」→「檔案上傳」→ 選擇作業壓縮檔(.zip)

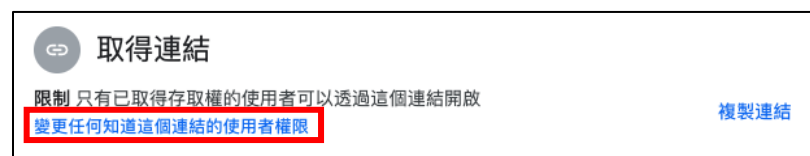


4. 開啟連結共用設定，並複製下載連結

1) 對檔案按右鍵，點選「共用」



2) 點擊「變更任何知道這個連結的使用者權限」後，權限會變為「可檢視」



3) 點選「複製連結」



5. 待加入課程臉書社團後，將連結上傳至作業貼文提供的「作業提交表單」



Should you have any idea or questions, please feel free to contact.